



Behavior Ontology and Modeling Patterns for Spacecraft Power Analysis

**Kenny Donahue, Seung Chung, Matt Rozek, Mitch
Ingham, John Day**

August 2013

**Jet Propulsion Laboratory, California Institute of Technology
Copyright 2013 California Institute of Technology
Government sponsorship acknowledged**



Background and Motivation

Timeline Concept

Timeline Ontology

SysML Modeling Patterns

SysML-Timeline-Maple Transformation

Questions

Behavior Modeling is an *ad-hoc* Process



- **Currently, there are languages and standards for representing behavior, but no real methodologies**
- **Behavior modeling differs from project to project and from analysis to analysis**
 - Behavior models cannot typically be shared between different tools/analyses because they contain analysis- or tool-specific information
- **Because of the lack of consistency/rigor, problems arise:**
 - Much of the behavior descriptions are up for interpretation
 - Many different sources for similar information lead to consistency issues

Benefits of having a formalized approach

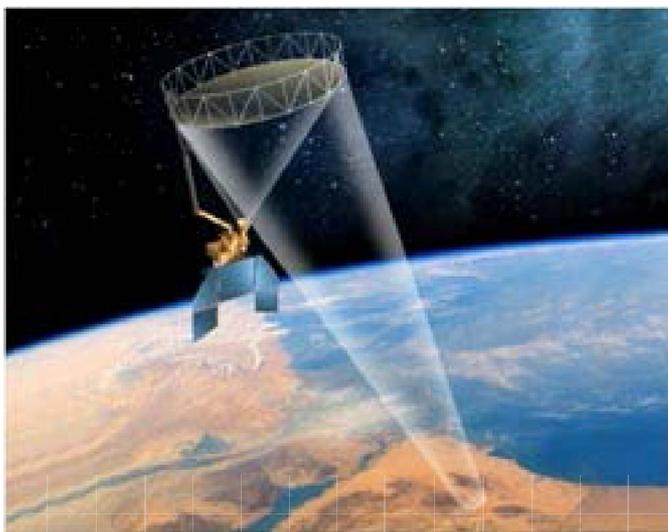
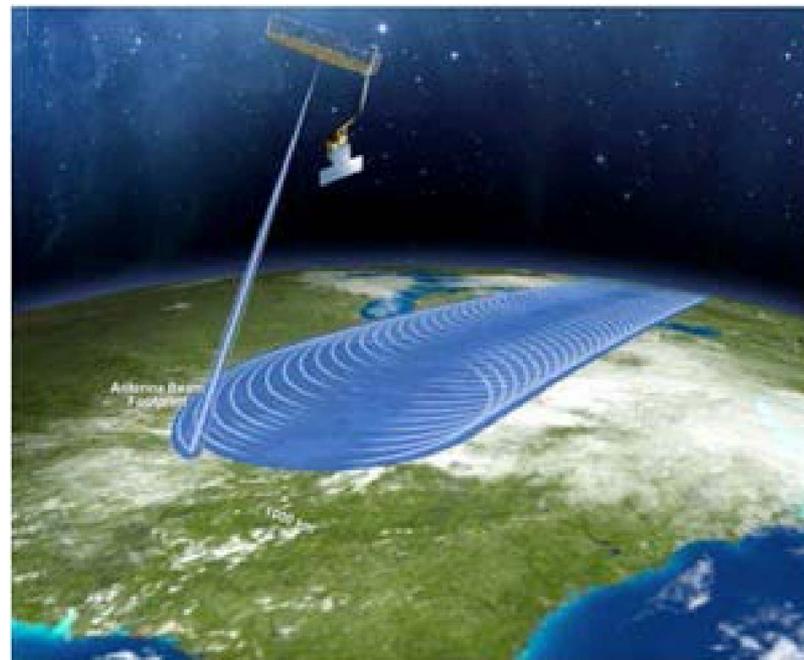


- **Improved Communication & Collaboration:**
 - Between systems team members, mission planners, subsystem engineers, etc...
 - Common source of information (“truth”) that can be continually maintained to be up to date
- **Improved Quality:**
 - Transformations are scripted, so human math / copy-paste / etc. errors are reduced
 - Scripts provide for some level of specification error checking (e.g. over or underconstrained definitions w.r.t. time)
 - Can help find inconsistencies within scenario definitions
- **Increased Productivity:**
 - Automated nature allows for analysis results to be generated/updated in the background without much human intervention
 - Less time for a higher quality product
- **Reduced Risk:**
 - Early and on-going scenario validation from a power loads POV

Mission Context: SMAP (Soil Moisture Active Passive)



- **One of four first-tier missions recommended by the Earth Science Decadal Survey**
- **Provide global mapping of soil moisture and freeze/thaw state to**
 - Link water, energy, and carbon cycles
 - Estimate water and energy fluxes at land surface
 - Quantify net carbon flux in boreal landscapes
 - Enhance weather and climate forecast skill
 - Develop improved flood prediction and drought monitoring capabilities

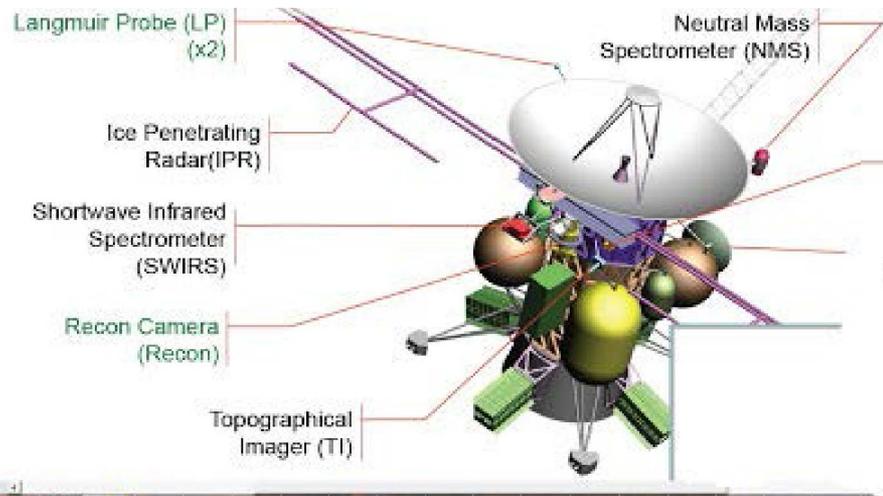
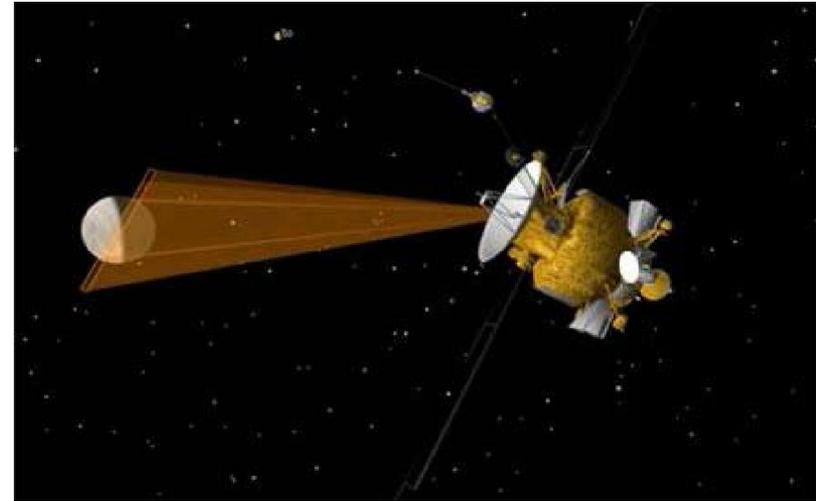


- **The Instrument suite includes:**
 - A Radiometer,
 - A Synthetic Aperture Radar (SAR),
 - A conically-scanning deployable mesh reflector
- **For this study, we focused on the power loads for various mission events**

Mission Context: Europa Clipper



- **A multi-flyby concept focusing on characterizing Europa**
 - Look for signs of liquid water below icy shell
 - Map surface features to decipher geological processes and activity
 - Characterize atmosphere composition
- **Highly-capable, radiation-tolerant spacecraft**
- **~30 flybys of Europa at altitudes varying from 2700 km to 25 km.**



- **The Instrument suite includes:**
 - Neutral Mass Spectrometer
 - ShortWave IR Spectrometer
 - Topographical Imager
 - Ice-Penetrating Radar
- **For this study, we focused on the power loads for various mission events**



Background and Motivation

Timeline Concept

Timeline Ontology

SysML Modeling Patterns

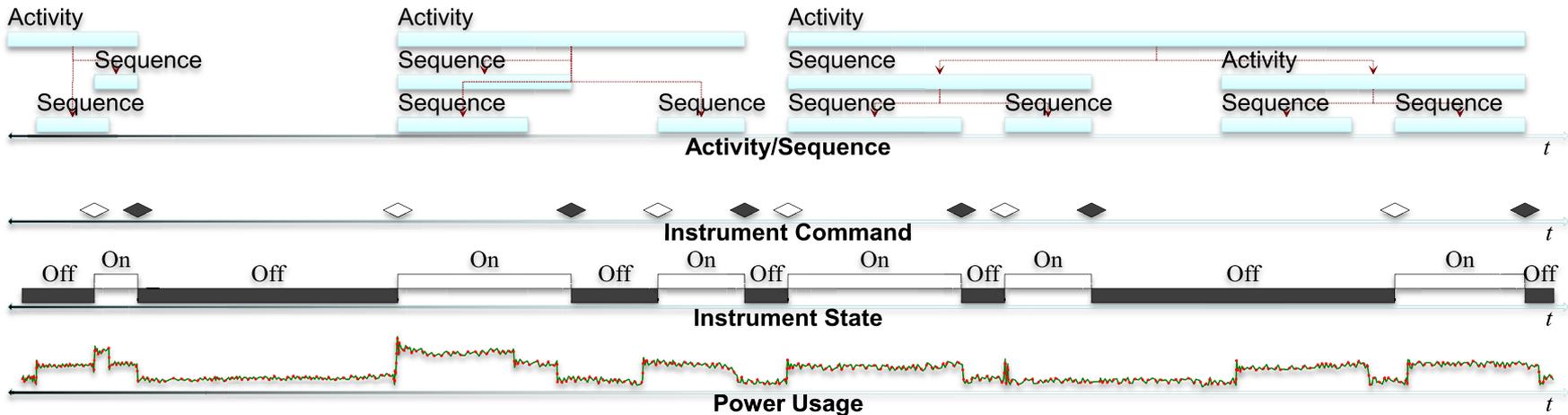
SysML-Timeline-Maple Transformation

Questions

Timeline Concept (1/3)



A timeline is a representation of time-varying information; more specifically, a representation of a set of *values* with associated times. The time domain of a timeline may be discrete or continuous. The times of the *values* can be assigned or be a variable. Additionally, the range of the variable times may be restricted by *temporal constraints*.

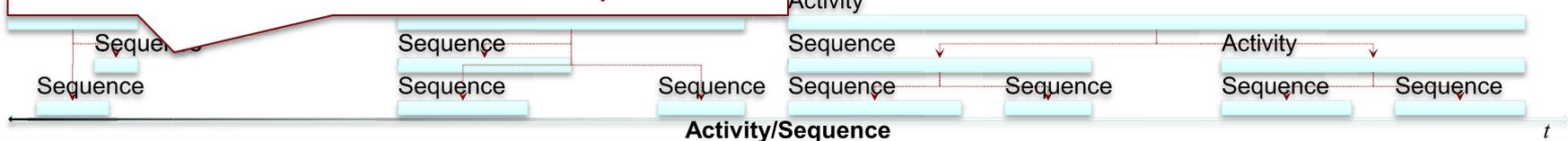


Timeline Concept (2/3)



A timeline value is a quantity (not necessarily numeric) belonging to the codomain specified for that timeline.

Codomain: Possible Activities/Sequences



Codomain: Possible Commands



Codomain: Possible Instrument States



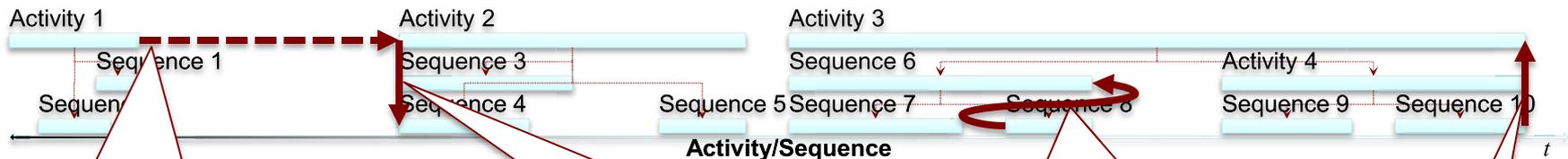
Codomain: Real [W]



Timeline Concept (3/3)



A temporal constraint is a restriction on the possible times for particular variable times.



**Activity 2 Starts
[0,3) min After
Activity 1 Ends**
 $0 \text{ min} \leq (A2_{\text{start}} - A1_{\text{end}}) < 3 \text{ min}$

**Sequence 4 Starts
Immediately When
Activity 2 Starts**
 $S4_{\text{start}} - A2_{\text{start}} = 0 \text{ min}$

**Sequence 6 Ends
(2,5] min After
Sequence 8 Starts**
 $2 \text{ min} < (S6_{\text{end}} - S8_{\text{start}}) \leq 5 \text{ min}$

**Activity 3 Ends
Immediately When
Sequence 10 Ends**
 $A3_{\text{end}} - S10_{\text{end}} = 0 \text{ min}$



Background and Motivation

Timeline Concept

Timeline Ontology

SysML Modeling Patterns

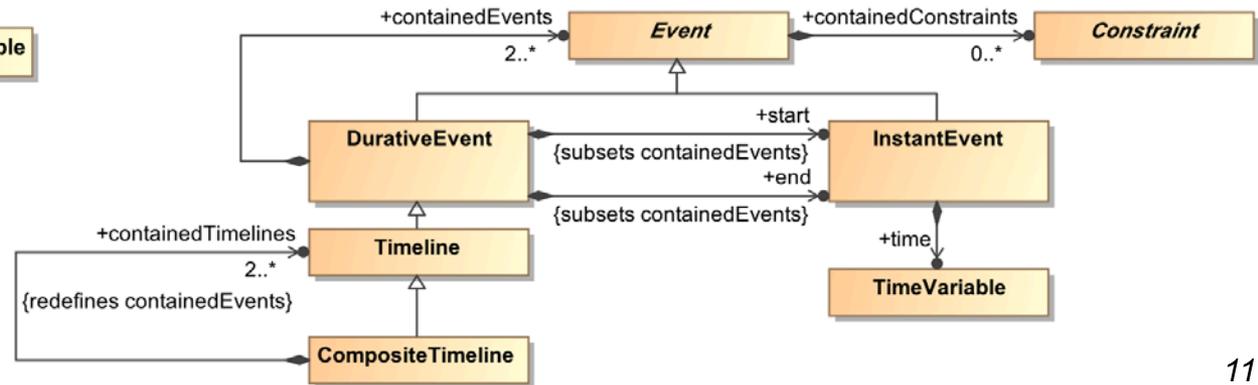
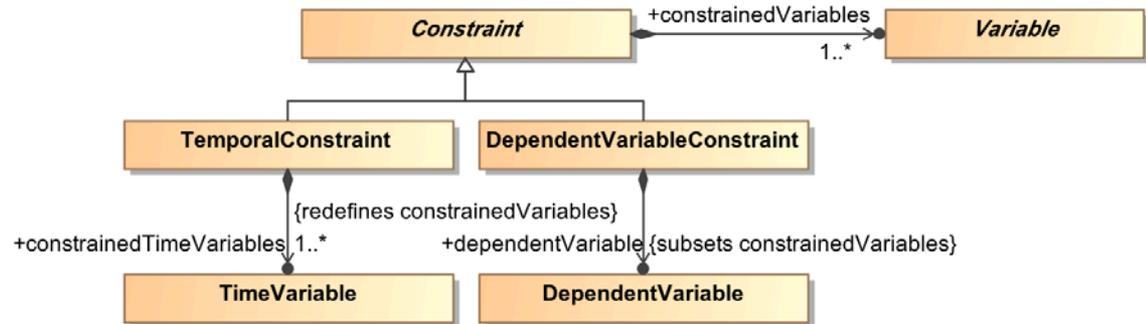
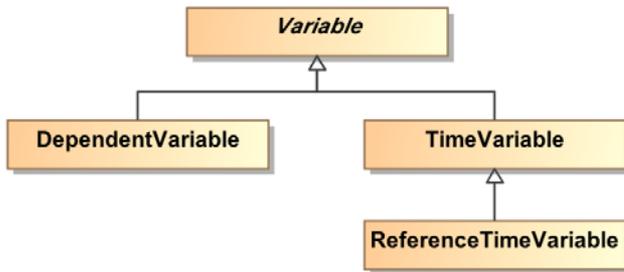
SysML-Timeline-Maple Transformation

Questions

Timeline Ontology (1/4) - Overview

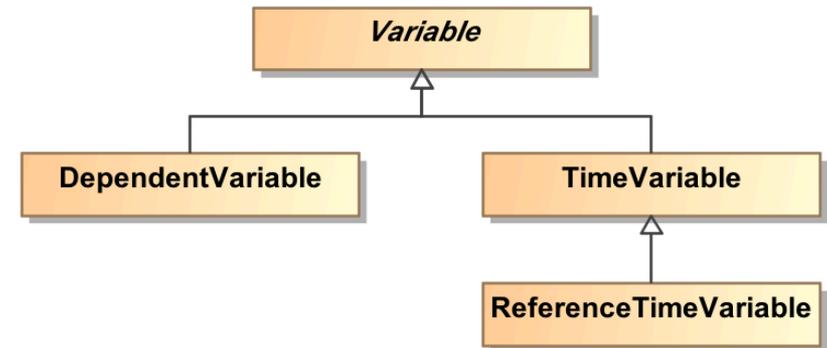


- With those concepts in mind, we defined an ontology in the Web Ontology Language (OWL) to give formal semantics to Timeline.
- OWL is a language for representing classes and their relationships formally.
- It is endorsed by W3C.





- **Variables: something to represent symbolically in our constraints**

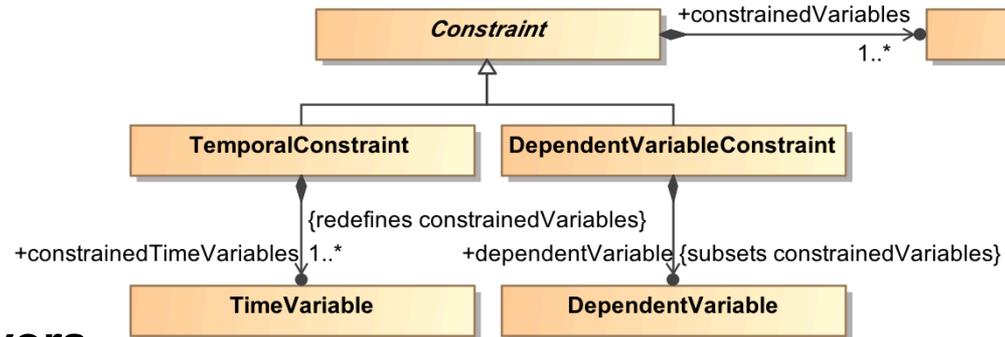


- **These come in three flavors**

- A *TimeVariable* is something that represents an instant in time (t_1 , t_2 , t_3 , etc). These are used to bound the temporal extent of constraints.
- A *ReferenceTimeVariable* is time in general (t). This serves as the reference frame for time. Currently, we assume there is only one of these per Timeline, but there could be multiple (e.g. $t_{\text{spacecraft}}$, $t_{\text{instrument}}$, t_{earth}).
- A *DependentVariable* is a handle for the property we are tracking over time (the *ReferenceTimeVariable*) (e.g. power, state, command).



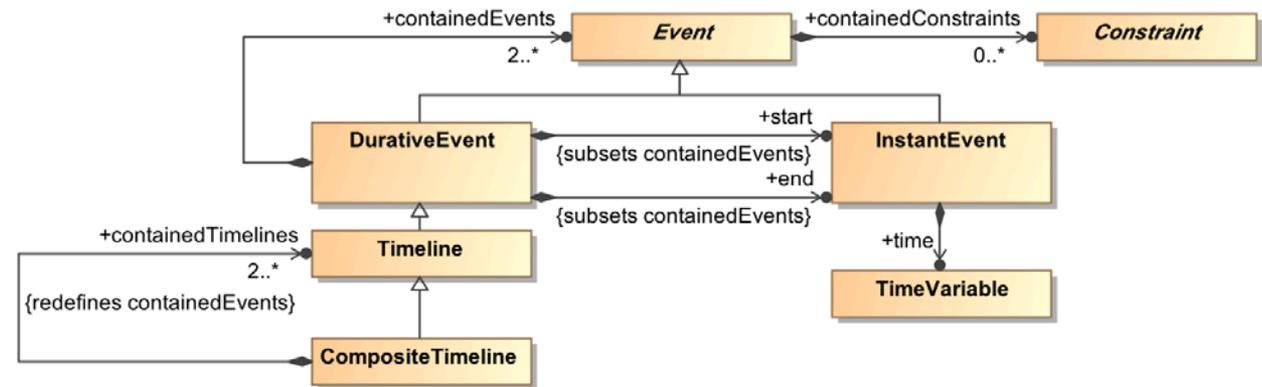
- **Constraints: something that restricts the value of a Variable**



- **These come in two flavors**

- A **TemporalConstraint** is something that constrains the value of only **TimeVariables** (e.g. $t_end = t_start + 4$, $t_end = 10$, $t1 > t2$, $t3 < 4$)
- A **DependentVariableConstraint** is something that constrains the value of a **DependentVariable** (e.g. $power[Component] = 10 * e^{-(t-t_start)}$)

- **Events: an interval over which a collection of constraints are valid**



- **These come in two main flavors**
 - An Instant Event is an event with no temporal extent (i.e. $t_{start} = t_{end}$). As such, it has a reference to a single instant in time (TimeVariable).
 - A DurativeEvent is an event with temporal extent (i.e. $t_{start} < t_{end}$). As such, it has a reference to a start InstantEvent and an end InstantEvent
 - Note: Because the Event gives the interval and it holds a collection of additional (DependentVariable)Constraints, the constraints of the event can be written in the following format:
 - $power[Component] = 10 * e^{-(t-t_{start})}$, $t_{start} < t < t_{end}$



Background and Motivation

Timeline Concept

Timeline Ontology

SysML Modeling Patterns

SysML-Timeline-Maple Transformation

Questions

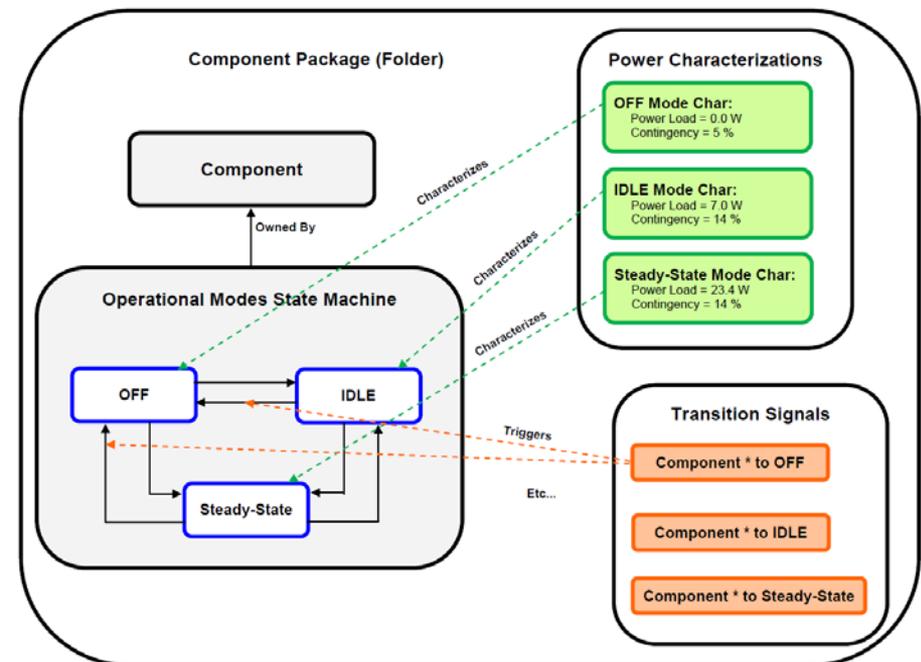
Behavior Specification:

Each Component needs to have a description of how it works. This should include nominal and off-nominal behavior (more than just a specification of intended behavior)

For this example, we use a SysML State Machine. A State Machine is a collection of States and Transitions between those States.

Each State needs to have a mathematical description of what the DependentVariables do in that State. We call these descriptions Characterizations.

A Characterization is a collection of DependentVariableConstraints.



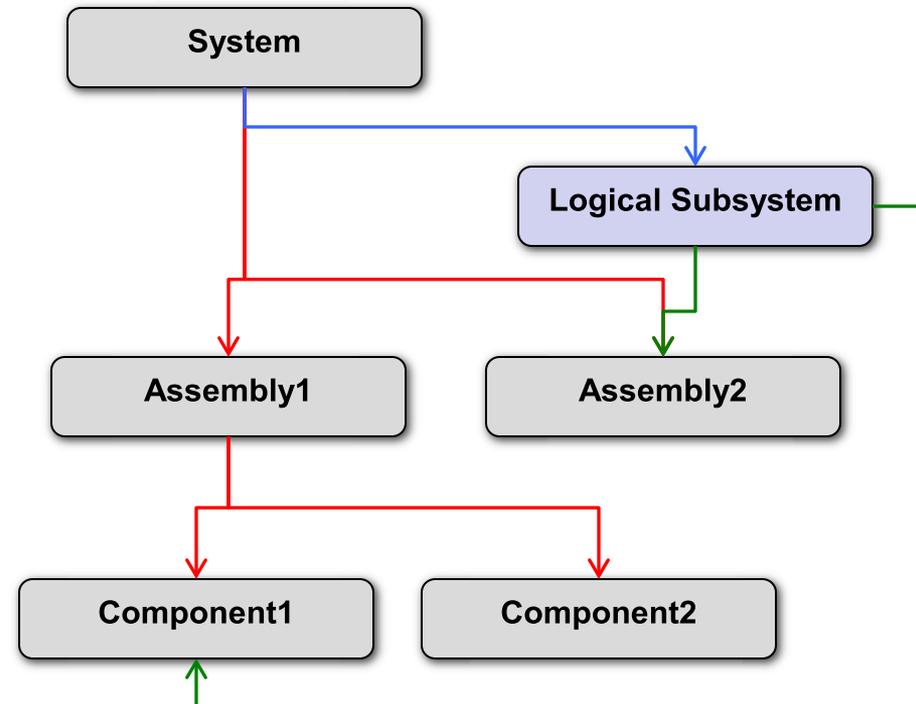


Component Hierarchy:

Each leaf-level Component is part of a larger System. The structure of that System needs to be well-defined in order to properly have leaf-level DependentVariable effects trickle up to the System-level.

Because there are many ways to slice a System, it is unlikely that only one Component Hierarchy will exist.

To ensure that we did not double-count leaf-level effects when trickling up, we created additional semantics to the directed association relationships to differentiate between physical and logical relationships. This is not strictly necessary, but having these distinctions makes the model more agile.



Scenario:

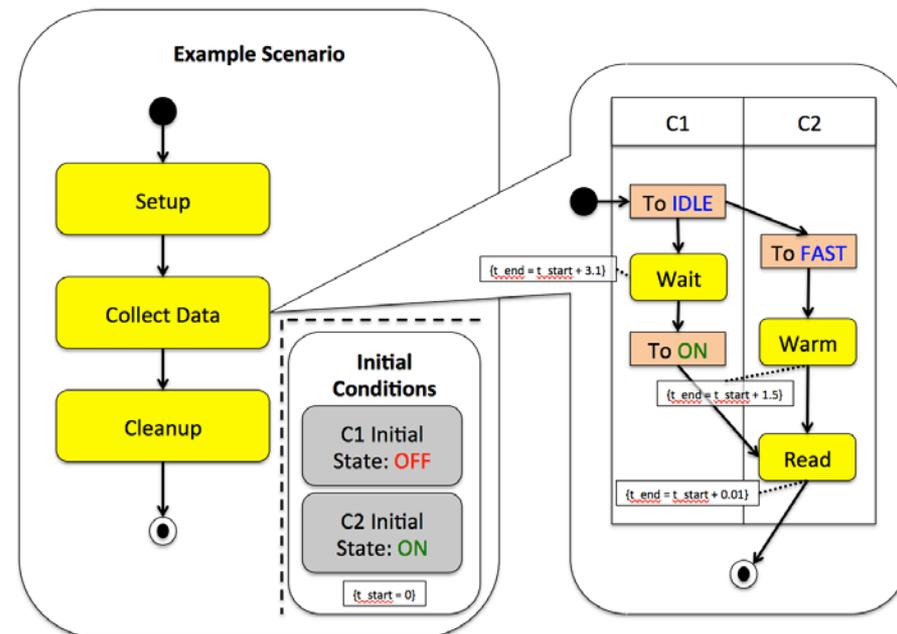
Once each Component's Behavior Specification has been defined, we can define sequences that trace then entire System through its State-space by controlling the state of each individual Component.

We use Activities for these Scenario representations, though Interaction/Sequence Diagrams would work equally well.

Components change states via AcceptEventAction with a Trigger that is a Signal. This signal should map to the Trigger of a Transition in the Component's Behavior Specification State Machine.

Actions can be CallBehaviorActions with an underlying Activity (making a Scenario a recursive structure).

ActivityEdges act as implicit TemporalConstraints. Additional TemporalConstraints can be provided explicitly.



Scenario Initial Conditions:

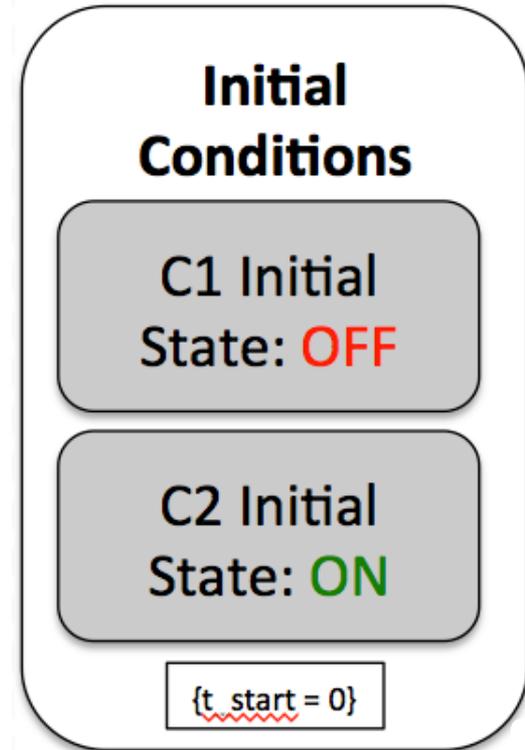
For a Scenario to work, certain key pieces of information must be provided

In order for a state transition to make sense, the initial State of each Component must be known. We do this with a special Characterization.

A start time must also be provided (some scenarios will want to start at non-zero times, like Launch, where many activities happen before the actual event)

The user additionally must specify the DependentVariables for the Scenario. There are likely many Characterizations on each State, so the user must provide which ones are of interest. (In this case, Power is assumed)

Once the DependentVariables have been selected, the “Rollup” method must be defined for each variable. (In this case, add())





Background and Motivation

Timeline Concept

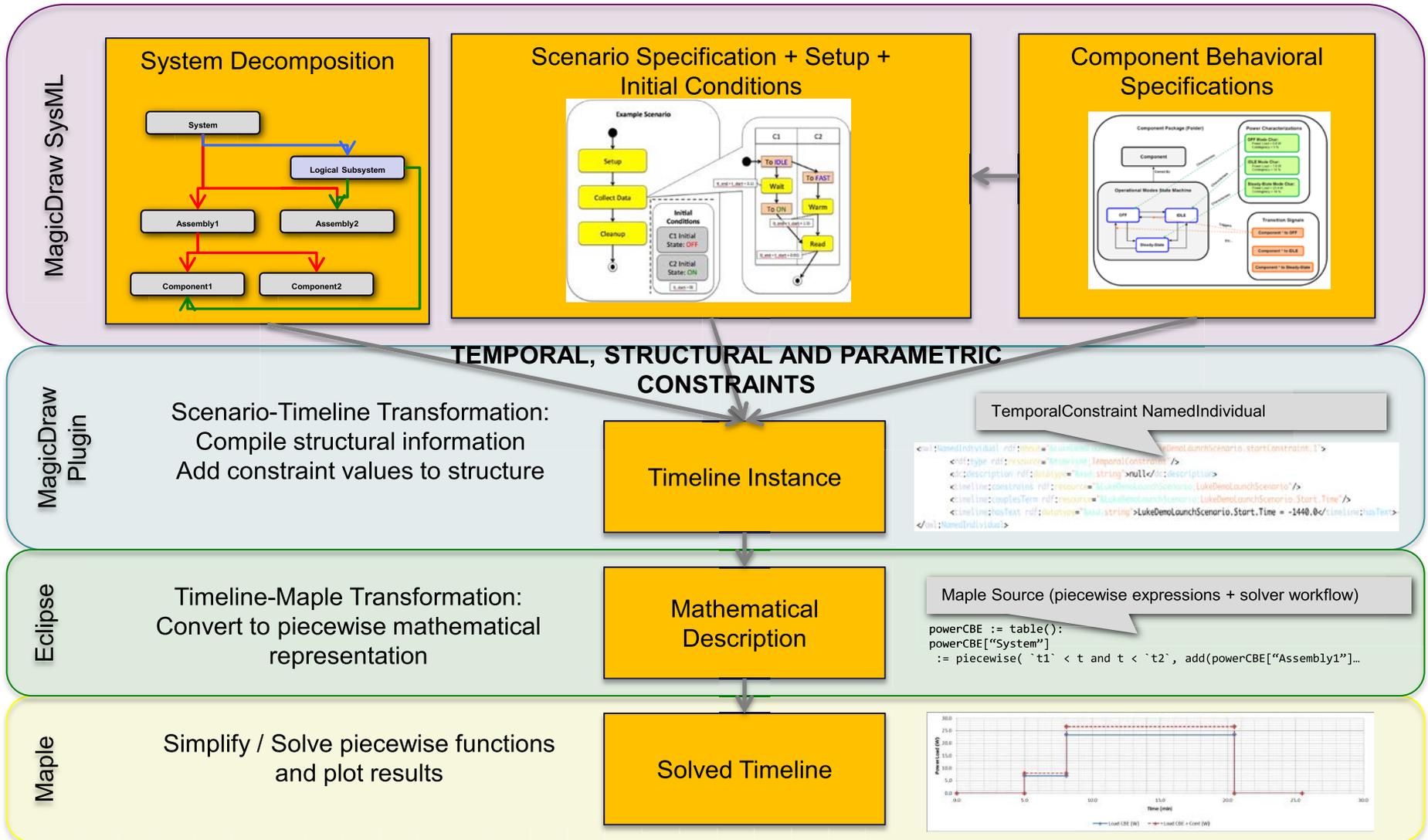
Timeline Ontology

SysML Modeling Patterns

SysML-Timeline-Maple Transformation

Questions

Transformations for Scenario Analysis (1/3)

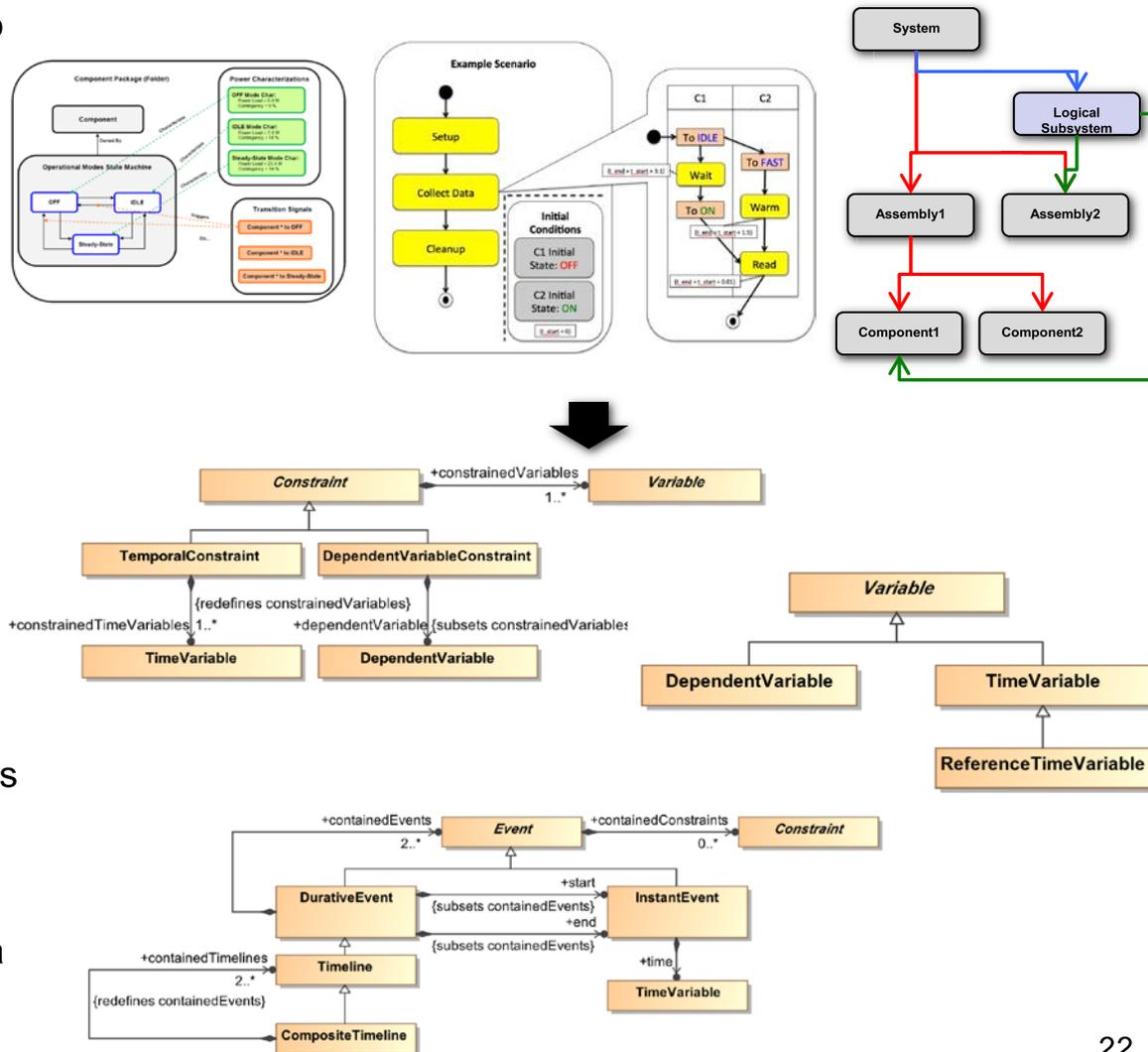


Transformations for Scenario Analysis (2/3)



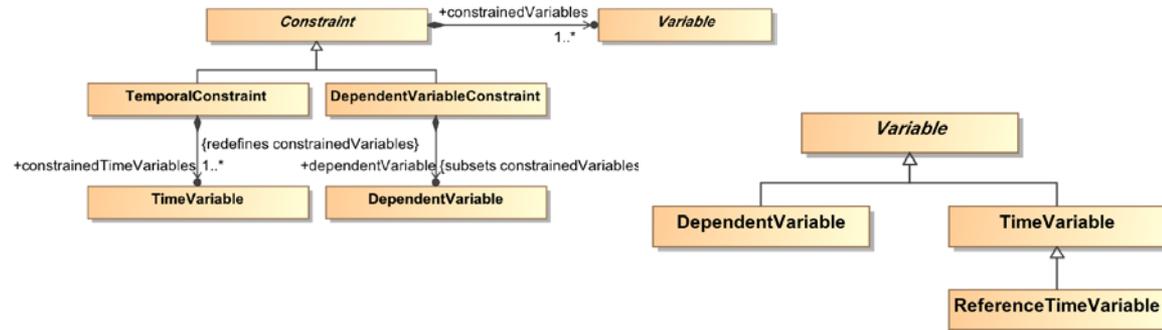
Scenario-to-Timeline Transformation

- Actions/Activities/States all map to DurativeEvents.
- Activities are recursive, but so are DurativeEvents.
- InitialNode, FinalNode, SendSignal, AcceptEventAction all map to InstantEvents.
- ActivityEdges map to Temporal Constraints.
- as do explicit constraints on Actions/Activities.
- Properties on Characterizations map to DependentVariables. Constraints on Characterizations map to DependentVariableConstraints.
- Hierarchy information maps to a tree of DependentVariableConstraints.

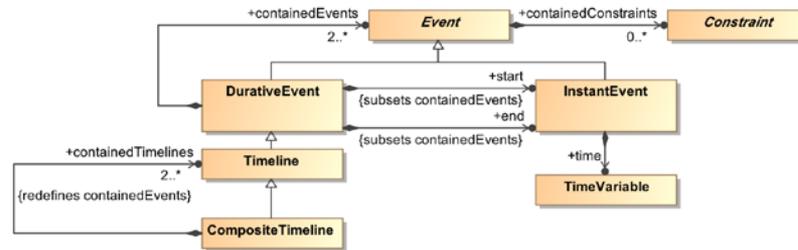




Timeline-to-MapleTransformation



- All the DurativeEvents have temporal extent, so all the constraints can be rewritten as piecewise expressions.
- (This transformation is significantly less involved)



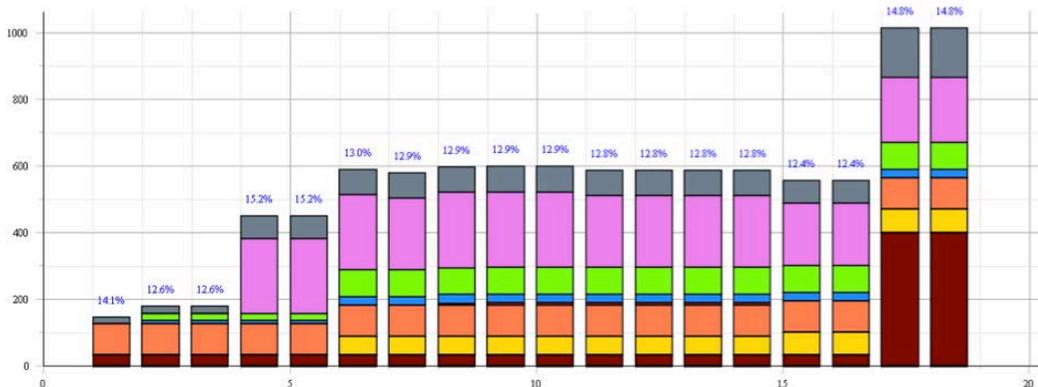
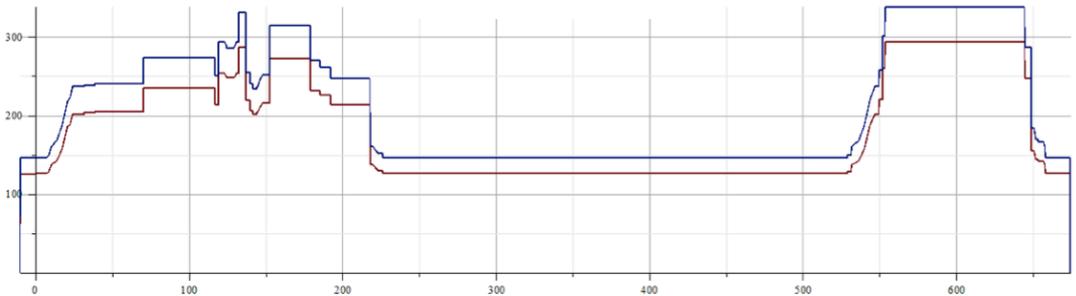
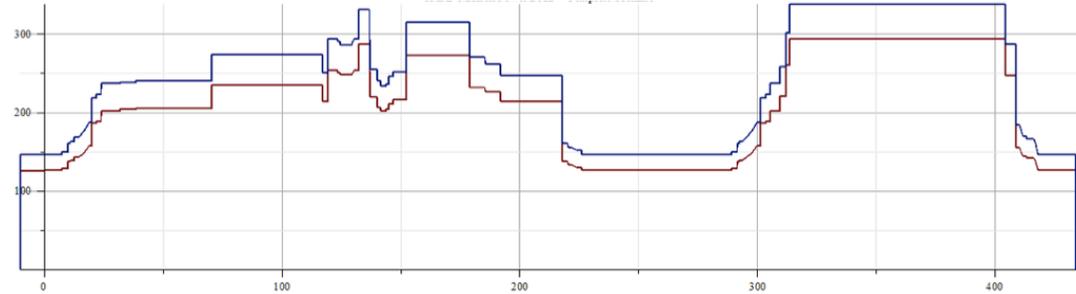
```

powerCBE := table():
powerCBE["System"]
:= piecewise( `t1` < t and t < `t2`, add(powerCBE["Assembly1"]...
    
```

Resulting Power Profiles



- We can plot the raw solutions in Maple (at any level in our component hierarchy)
- Underconstrained timelines are really parameterized, making it easy to see the effect of different durations, power loads, etc.



- With a little post-processing in Maple, we can use the raw solutions to produce composite plots like this that show subsystem-by-subsystem breakdown of power loads over time.



Background and Motivation

Timeline Concept

Timeline Ontology

SysML Modeling Patterns

SysML-Timeline-Maple Transformation

Questions

Acknowledgements



This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Questions?

